CLAIMS

1. Method for operating (implementing) a secondary operating system on a computer in addition to a primary operating system, wherein a secondary operating system driver (SOS driver) of the primary operating system is loaded for loading and controlling the secondary operating system.

2. Method according to claim 1, wherein the secondary operating system driver (SOS driver) subsequently loads the secondary operating system.

3. Method according to claim 1 or 2, wherein the secondary operating system driver loads the secondary operating system.

4. Method according to one of the claims 1 to 3, wherein memory contexts (virtual operating areas) are created in the central processing unit (CPU).

5. Method according to one of the claims 1 to 4, wherein a change between the operating systems takes place by means of the SOS driver of the primary operating system and the board support package (BSP).

6. Method, particularly according to one of the preceding claims, wherein there is an exchange of interrupt tables on changing the dependence of the operating systems.

7. Method according to one of the claims 1 to 6, wherein the secondary operating system controls a change to the primary operating system.

8. Method according to claim 7, wherein a change from the secondary operating system to the primary operating system takes place when the secondary operating system is idle (entry into the idle loop).

9. Method according to claim 8, wherein a change from the secondary operating system to the primary operating system takes place through an instruction in the program sequence of the secondary operating system.

10. Method according to one of the claims 1 to 9, wherein a change from the primary operating system to the secondary operating system takes place through an interrupt call.

11. Method according to one of the preceding claims, wherein the change between operating systems takes place by means of a program code filed in the tunnel area of the memory.

12. Method according to one of the preceding claims, wherein interrupt calls of the primary operating system are inhibited during the secondary operating system sequence.

13. Method according to one of the preceding claims, wherein an interrupt servicing routine in the SOS operator reads the interrupt call table of the secondary operating

system and the processing of the latter takes place or is continued at the point relative to the interrupt call.

14. Method according to one of the preceding claims, wherein for each interrupt associated with the secondary operating system (i.e. which is to initiate an interrupt call in the secondary operating system) the system driver generates an entry in the interrupt call table in the primary operating system, which in turn initiates a call of the corresponding interrupt servicing routine in the secondary operating system.

15. Method according to one of the preceding claims, wherein by means of an interrupt call servicing routine in the system driver, the information stored in the interrupt table of the secondary operating system (SOS) is determined as to the point in the latter where the running of the interrupt is to take place.

16. Method according to one of the preceding claims, wherein in the case of activity of the secondary operating system (SOS) following an interrupt request through the information stored in the interrupt call table of the secondary operating system as to the point in the latter where the running of the interrupt is to take place, the interrupt call servicing routine of the secondary operating system (SOS) is directly polled solely by the secondary operating system and not via the system driver.

17. Method according to one of the claims 12 to 16, wherein after occurence a corresponding interrupt call and determi-

nation of the point in the secondary operating system where interrupt running is to take place is determined, processing thereof at the point in the secondary operating system concerning the interrupt call is continued.

18. Method according to one of the preceding claims, wherein on changing from one operating system to the other all the system states of one operating system are stored.

19. Method according to one of the preceding claims, wherein on changing from one operating system to the other all system states of the other operating system are loaded.

20. Method according to one of the preceding claims, wherein clock generation for the secondary operating system takes place through the hardware timer.

21. Method according to one of the preceding claims, wherein clock generation for the primary operating system takes place through a clock system driver.

22. Device for operating a secondary operating system on a computer in addition to a primary operating system, wherein a secondary operating system driver (SOS driver) of the primary operating system for loading and controlling the secondary operating system is constructed.

23. Device according to claim 22, wherein the SOS driver has a tunnel context setting routine for setting a tunnel context in the central processing unit (CPU).

24. Device, particularly according to claim 22 or 23, wherein said device is constructed for exchanging interrupt tables on a change of activity of the operating systems.

25. Device according to claim 22 or 24, wherein the SOS driver has an interrupt call table change routine for producing entries in the interrupt call table of the primary operating system, which at least take up entries for the interrupt calls for the secondary operating system.

26. Device according to one of the claims 22 to 25, wherein the board support package (BSP) has a section for return to the primary operating system (POS).

27. Device according to one of the claims 22 to 26, wherein the secondary operating system driver (SOS driver) has an interrupt table section by means of which it produces in the primary operating system an interrupt call table containing a call of an interrupt servicing routine for polling the secondary operating system.

28. Device according to one of the claims 22 to 27, wherein the system driver is constructed for producing an entry in the interrupt call table in the primary operating system (POS) for each interrupt associated with the secondary operating system (SOS), which i.e. is intended to initiate an interrupt call in the secondary operating system and that the interrupt call table is constructed for polling the corresponding interrupt servicing routine in the secondary operating system (SOS).

29. Device according to one of the claims 22 to 28, wherein an interrupt call servicing routine in the system driver is constructed for determining the information stored in the SOS interrupt table as to the point in the secondary operating system where interrupt running is to take place.

30. Device according to one of the claims 22 to 29, wherein it is constructed in the case of activity of the secondary operating system (SOS) following an interrupt request through the information stored in the secondary operating system interrupt call table as to the point in which the secondary operating system interrupt running is to take place, so as to poll the interrupt call servicing routine of the secondary operating system directly solely through the secondary operating system and without passing via the system driver.